MODÜL 7

FONKSİYONLAR, GLOBAL VE LOKAL DEĞİŞKENLER

Şekil 7.1: Bölümle ilgili örnek uygulamalara karekoddan ulaşabilirsiniz

7.1. Fonksiyon Kullanımı

Bu bölüme kadar kullanılan fonksiyonlardan bazıları şunlardır: print( ), input( ), int( ) ve len( )

Fonksiyonlar, matematikteki fonksiyonlarla aynı işlevi görürler. y=f(x)=x*x gibi bir fonksiyon tanımlandığında verilen değeri kendisiyle çarpar. Bu görece kolay bir işlemdir. Bir sayının karesini hesaplamak gerektiğinde, formülü tekrar yazmak yerine bu fonksiyon kullanılabilir. Fonksiyonlar; kodları tekrar yazmayı ortadan kaldırmak, kodları daha iyi organize etmek sonrasında bu kodları rahat bir şekilde kullanmak için oluşturulur.

Python'da ve diğer programlama dillerinde birçok fonksiyon çeşitli kütüphanelerde veya varsayılan olarak tanımlı gelir. Bu sayede en temel işlemler için bile satırlarca kodu tekrarlamaya gerek kalmaz.

Fonksiyonların görevi, karmaşık işlemleri bir araya toplayarak bu işlemleri tek adımda yapmayı sağlamaktır. Fonksiyonları kullanarak bir veya birkaç adımdan oluşan işlemleri tek bir yapı altında toplamak mümkündür.

Fonksiyonun Adı	Parametreler - Argümanlar
print	(“Sultan”, “Murat”, sep=“ ”, end=“\n”)
input	(“Argüman”)

Her fonksiyonun bir adı (print, input, len vb.) ve işlevini yerine getirmesi için kullanabileceği parametre adı verilen yapıları vardır. Parametreler fonksiyonda tanımlı özellik ve girdileri belirtir. Bir fonksiyonu kullanırken (çağırırken) bu parametrelere verilen değerlere “argüman” denir. Aşağıda print( ) fonksiyonu ve parametrelerinin kullanımına örnek verilmiştir. “end” ve “sep” birer parametredir. Fonksiyonlarda bazı parametreler zorunludur ve bir argüman (değer) girmek gerekir. Ancak yine bazı parametreler fonksiyon içinde ön tanımlı olarak bir değer/boş (None) veya seçmeli olarak belirlenmiştir. Bu parametrelere değer vermek gerekmez. Bunlar, isteğe bağlıdır ve kullanılmayacak özellikler için argüman girmeye gerek yoktur.

Örnek 1

Aşağıdaki örnekte print ( ) fonksiyonun “end” ve “sep” argümanlarıyla birlikte kullanımı gösterilmiştir:

```
print ('Merhaba', 'Mars', sep=' ', end='\n') #parametreleri kullanarak
print ('Merhaba')
print ('Merhaba',' ', 'Mars') #parametreleri kullanmadan
print ('Merhaba')
Merhaba Mars
Merhaba
Merhaba   Mars
Merhaba
```

7.2. Fonksiyon Oluşturma

Temel fonksiyonlar, Python içinde hazır olarak yer alan fonksiyonlardır. Bu fonksiyonlar tanımlı olduğu için sadece fonksiyonadı( ) yazarak kullanabilmektedir.

Bir fonksiyon def fonksiyonAdi(parametre1, parametre2,..):şeklinde tanımlanır. Fonksiyon içindeki kodlar girinti şeklinde blok yapısında yazılır. Fonksiyon yapısı dışında o fonksiyonu kullanmak için fonksiyonAdi(argüman1, argüman2) şeklinde parametre değerleri verilerek kullanılır. Örneğin, girilen bir sayının çift bir sayı olup olmadığını bulan bir fonksiyon yazabilirsiniz. Fonksiyon bir parametreden oluşacak ve bir sayı değeri alacaktır.

Örnek 2

Bir sayının çift olup olmadığını ekrana yazan bir fonksiyon.

Hatırlatma: Bir sayının 2'ye bölümünden kalan yoksa o sayı çifttir varsa tektir.

```
def sayiCiftMi (sayi):
  if sayi%2==0:
    print('Sayı çifttir')
  else: ('Sayı tektir')
```

Bir fonksiyonu oluşturduktan sonra print( ) fonksiyonunda kullandığınız şekilde parametrelere uygun argümanları girip çağırabilirsiniz.

```
sayiCiftMi(10)
Sayı çifttir
```

Fonksiyonun adı ve parametrelerine değerleri yazılarak fonksiyon kullanılabilir. Fonksiyon çağrıldığında verilen argümanlara göre işlem yapar. Örnekteki fonksiyon 10 sayısını kontrol ederek sonucu "Sayı çifttir" şeklinde ekrana yazdırır.

Örnek 3

Başka bir fonksiyon tanımlayabilirsiniz. Tanımlanan fonksiyon bir metni istenildiği kadar alt alta yazdırır. Bu fonksiyonda yazdırılacak metin ve yazdırılma sayısı olmak üzere iki adet parametre olacaktır.

```
def yazdir(metin,kacKere):
  for i in range (1, (kacKere+1)):
    print (metin, end='\n')
#Fonksiyon çağırma
yazdir('Merhaba', 5)
Merhaba
Merhaba
Merhaba
Merhaba
Merhaba
```

Aynı fonksiyon için parametreleri kullanıcıdan alabilirsiniz.

```
yazdirilacakMetin=input('Yazdırılacak metni giriniz: ')
yazdirmaSayisi=int(input('Metin kaç kez yazdırılacak: '))
yazdir(yazdirilacakMetin, yazdirmaSayisi)
Yazdırılacak metni giriniz: Kara Murat
Metin kaç kez yazdırılacak: 4
Kara Murat
Kara Murat
Kara Murat
Kara Murat
```

Örnek 4

Bir sayının asal bir sayı olup olmadığını bulan bir fonksiyon yazabilirsiniz. Fonksiyon, sayı asal ise "True"; değilse "False" değerini döndürecektir.

NOT

"for" döngüsünde sayının yarısına kadar bakmamızın nedeni bir sayının kendi değerinin yarısından önce böleni yok ise sonrasında da olmayacağı kuralıdır.

```
def asalMi(sayi):
  sayac=2 # tüm sayılar 1'e bölüneceğinden 2 ile başlattık
  while sayac<=int(sayi/2):
    if sayi%sayac==0:
      return False
    sayac+=1
  return True
#Fonksiyonu çağırma
asalMi(113)
True
```

Örnek 5

Verilen sayının faktöriyelini bulan bir fonksiyon tanımlayabilirsiniz. Bir sayının faktöriyeli kendisinden başlayarak 1'e kadar olan tüm sayıların çarpımıdır. 3!=3*2*1)

```
def faktoriyelAl(sayi):
  sonuc=1
  if  (sayi==0 or sayi==1):
    print('sonuç= ', 1)
  elif sayi>1:
    for i in range(1, sayi+1, 1):
      sonuc*=i
    print ('sonuc=', sonuc)
  else: print ('0 veya daha büyük sayısal bir değer girmelisiniz')
```

Yukarıda tanımlanan fonksiyonu çağırmak için kullanıcıdan argüman almanız gerekmektedir.

```
faktoriyelAl(int(input('Faktöriyeli alınacak sayıyı giriniz: ')))
Faktöriyeli alınacak sayıyı giriniz: 5
sonuc= 120
```

7.3. Fonksiyonlarda Parametre Türleri

"faktoriyelAl" olarak tanımlanan fonksiyon "sayi" parametresiyle çalışmaktadır. Argüman olarak bir sayı verildiğinde sayının faktöriyelini hesaplamaktadır. Fonksiyonlar yaptıkları işlemlere göre farklı parametre türlerini kullanır.

Örnek 6

Döngülerde örnek olarak verilen ağaç çizme kodlarını bir fonksiyon hâline getirebilirsiniz.

```
def agacCiz(agacinYuksekligi, karakter='*'):
  b=agacinYuksekligi
  for i in range(1,agacinYuksekligi+1):
    print(b*' ',(2*i-1)*karakter)
    b-=1
```

Şimdi, fonksiyonu kullanıcıdan aldığınız parametrelerle çağırabilirsiniz.

```
agacYuksekligi=int(input("Ağacın yüksekliği kaç satır olsun? : "))
agacKarakteri=input("Ağaç için bir sembol veya karakter girin? : ")
if agacKarakteri!='' and agacYuksekligi>=1:
  agacCiz(agacYuksekligi, agacKarakteri[0])
elif agacKarakteri=='' and agacYuksekligi>=1:
  agacCiz(agacYuksekligi)
else: print('Hatalı giriş')
```

```
Ağacın yüksekliği kaç satır olsun? : 10
Ağaç için bir sembol veya karakter giriniz :
           *
          ***
         *****
        *******
       *********
      ***********
     *************
    ***************
   *****************
  *******************
```

Yukarıda bir "agacCiz" fonksiyonu tanımlanmıştır. Bu fonksiyonun "agacinyuksekligi" ve karakter adlarında iki adet parametresi bulunmaktadır. "agacinYuksekligi" ağacın satır sayısını belirten bir parametredir. Bu parametreye bir argüman (değer) verilmezse fonksiyon hata verir. Böyle parametrelere "zorunlu parametre" denir. Karakter parametresi ise parametre olarak tanımlanırken bir değerle birlikte tanımlanmıştır, bu değer parametrenin varsayılan değeridir. Varsayılan değer olarak "None" boş değer verilebilir. Zorunlu parametreler dışında fonksiyonlarda kullanılan diğer parametreler için varsayılan değerler tanımlanabilir. Eğer kullanıcı karakter olarak bir giriş yapmazsa yani boş bırakırsa fonksiyon ağacı varsayılan olarak "*" karakterini kullanarak oluşturur.

7.4. Değer Döndüren ve Döndürmeyen Fonksiyonlar

Python'da fonksiyonlardan bazıları sadece bir işlevi yerine getirir ve görevi orada biter. Ama bazı fonksiyonlar bir değer döndürür. "int( )" fonksiyonu verilen argümanı sayıya çevirerek döndürür. Yukarıdaki örneklerde "faktoriyelAl" ve "agacCiz" fonksiyonları bir değer döndürmemektedir. Değer döndüren fonksiyonlarda "return" ifadesi yer alır. Bir fonksiyonun sadece ekrana yazı yazdırması değer döndürdüğü anlamına gelmez. Eğer bir fonksiyonun çıktısı uygun bir değişkene atanabiliyorsa bu fonksiyon değer döndüren bir fonksiyondur.

Örnek 7

Örnekte kullanılan faktoriyelAl() fonksiyonunu değer döndüren bir fonksiyon hâline getirebilirsiniz.

```
def faktoriyelAl(sayi):
  sonuc=1
  if  (sayi==0 or sayi==1):
    sonuc=1
  elif sayi>1:
    for i in range(1, sayi+1, 1):
      sonuc*=i
  else: sonuc=-1 #hatalı bir işlem olduğunu anlamak için -1 değerini veriyoruz
  return sonuc
```

Değer döndüren fonksiyonu aşağıdaki gibi çağırabilirsiniz.

```
sonucumuz=faktoriyelAl(5)
#fonksiyonu bir değişkene atayabiliyoruz.
if sonucumuz!=-1: # bir hata olup olmadığını kontrol edelim
  print(sonucumuz)
else:print('Bir hata oluştu')
120
```

Kendi tanımladığınız fonksiyonların içinde temel fonksiyonları ya da tanımladığınız başka fonksiyonları da kullanabilirsiniz.

Örnek 8

Kullanıcıdan liste olarak aldığı sayıların ortalamasını bulan bir fonksiyon tanımlayabilir ve bunun içinde len( ) fonksiyonunu çağırabilirsiniz.

```
def ortalamaBul(sayilar):
  sayilarinToplami=0
  sayilarinOrtalamasi=0
  for i in range(len(sayilar)):
    sayilarinToplami+=sayilar[i]
  sayilarinOrtalamasi=sayilarinToplami/len(sayilar)
  return sayilarinOrtalamasi
```

Sayıları kullanıcıdan alarak fonksiyonunuzu çağırabilirsiniz.

```
sayiAdedi=int(input('Kaç adet sayının ortalamasını alacaksınız: '))
sayilarim=[]
for i in range(0,sayiAdedi):
  print (i+1, '. sayıyı giriniz?')
  # indis sıfırdan başladığı için +1 kullandık
  sayi=int(input())
  sayilarim.append(sayi)
ortalamaBul(sayilarim)
Kaç adet sayının ortalamasını alacaksınız: 4
1 . sayıyı giriniz?
10
2 . sayıyı giriniz?
20
3 . sayıyı giriniz?
30
4 . sayıyı giriniz?
40
25.0
```

NOT

Fonksiyonlar bir fonksiyon da dâhil olmak üzere değer olarak her türlü veri tipini döndürebilirler.

7.5. Global ve Lokal Değişkenler

Python'da blok yapısından ve girintilerden söz etmiştik. Python'da tanımlanan ve değer atanan değişkenler tanımlandıkları blok içinde geçerlidir. Bir değişken aynı düzeydeki veya daha içerideki girintilerde de değerini korur.

Örnek 9

```
for i in range (1,3):
  print ('i değişkenin değeri=', i)
print ('i değişkenin son değeri=', i)
i değişkenin değeri= 1
i değişkenin değeri= 2
i değişkenin son değeri= 2
```

"for" bloğu en dışta yer aldığı için i değeri onunla aynı hizada ve daha içerideki girintilerde tanınmakta ve değerini korumaktadır. Ancak sadece iç girintideki bir blokta değer atanan değişken dış bloklarda tanımsızdır. Bu tür değişkenlere yani kendi bloğu ve daha içerideki girintilerde tanınan değişkenlere yerel değişken denir. Bu özellik program yazarken dikkat edilmesi gereken bir durumdur.

Örnek 10

"if" bloğu içinde tanımlanan bir değişkene erişebilirsiniz.

```
yas=35
if yas ==35:
  deger='yolun yarısı'
print (deger)
yolun yarısı
```

Örnek 11

Şart sağlanmazsa "if" bloğunun içindeki kodlar çalışmayacaktır. Örnekte bunu deneyebilirsiniz.

```
yas=34
if yas ==35:
  deger2='yolun yarısı'
print (deger2)
NameError                 Traceback (most recent call last)
<ipython-input-126-3f455a188ff0> in <module>()
      2 if yas ==35:
      3 deger2='yolun yarısı'
----> 4 print (deger2)
```

Hata mesajında deger2 adında bir değişkenin tanımlı olmadığı ifade edilmektedir.

En dışta tanımlanan (girintisiz) değişkenler global değişkenlerdir. Global değişkenler ilk değerlerini tüm alt bloklara taşırlar.

Örnek 12

Alt bloklarda (iç girintide) aynı değişkene farklı bir değer atanabilir.

```
yas=34 # global bir değişken
dogumGunuMu=True
if dogumGunuMu==True:
  yas+=1 #yerelde aynı değişken 1 artırılmıştır.
  print ('Nice yıllara! Yaş:', yas)
Nice yıllara! Yaş: 35
```

Örnek 13

if koşulu sağlanamazsa bloğun içindeki kodlar atlanır. Global değişken olarak tanımlanan yas2 değerini korur.

```
yas2=34 # global bir değişken
dogumGunuMu=False
if dogumGunuMu==True:
  yas2+=1 #yerelde aynı değişken 1 artırılmıştır.
  print ('Nice yıllara! Yaş:', yas2)
print ('Yaşınız: ', yas2)
Yaşınız:  34
```

Örnek 14

Aynı şekilde fonksiyonlar içinde tanımlanmış değişkenler yerel (lokal) değişkenler olduğu için fonksiyon çağrılmazsa hata ile karşılaşılır.

```
def toplamBul (sayiListesi):
  toplam=0
  for i in range (len(sayiListesi)):
    topla+=sayiListesi[i]
  return topla
print (topla)
NameError                 Traceback (most recent call last)
<ipython-input-132-e72b6ce4bac6> in <module>()
----> 1 print (topla)
NameError: name 'topla' is not defined
```

Örnek 15

Programınızda global bir değişken yazarak işlemi tekrarlayabilirsiniz.

```
topla=0
def toplamBul (sayiListesi):
  topla=0
  for i in range (len(sayiListesi)):
    topla+=sayiListesi[i]
  return topla
```

Fonksiyonu tanımlanmasına rağmen global değişkene atanan değer ekrana yazdırılabilir.

```
print (topla)
0
```

Şimdi de fonksiyonunuzu çağırabilirsiniz.

```
toplamBul([1, 2, 3, 4, 5])
15
```

Değişkene atanan yeni değer yani fonksiyonun sonucu görülmektedir.

7.6. Bölüm Sonu Örnekleri

1. Vize, final puanlarını ve yüzdelik oranlarını parametre olarak alan ve puanlar ile yüzdelik oranları çarparak toplayan ve sonucu döndüren bir fonksiyon tanımlayınız ve (50, 60, 30, 70) argümanlarıyla fonksiyonunuzu kullanınız. kullanın. Sonuç sizce kaç olur?

2. Bir liste halinde verilen sayılardan tek olanların toplamını döndüren bir fonksiyon yazınız. [1, 2, 3, 4, 5, 6, 7, 9, 11, 1773, 1679] sayı listesiyle deneyiniz. Sonuç sizce kaç olur?

3. Aşağıdaki kodun ekran çıktısı sizce nasıl olur?

```
karsilamaMesaji='Merhaba'
kullaniciYasi=35
dogumGunuMu=True
if kullaniciYasi==35 and dogumGunuMu==True:
  karsilamaMesaji1='Yolun yarısı'
  kullaniciYasi+=1
  print(dogumGunuMu)
dogumGunuMu=False
print (kullaniciYasi)
print(dogumGunuMu)
print(karsilamaMesaji)
```

Cevaplar

1.
```
def puanHesaplama (vizePuani, finalPuani, vizeYuzdesi, finalYuzdesi):
  return(vizePuani*vizeYuzdesi/100+finalPuani*finalYuzdesi/100 )
puanHesaplama(50, 60, 30, 70)
57.0
```

2.
```
def tekSayilariTopla(sayilar):
  sayilarinToplami=0
  for i in range(len(sayilar)):
    if sayilar[i]%2!=0:
      sayilarinToplami+=sayilar[i]
  return sayilarinToplami
tekSayilariTopla([1, 2, 3, 4, 5, 6, 7, 9, 11, 1773, 1679])
3488
```

3.
```
True
36
False
Merhaba
```